encapsulates the capabilities of multiple datastores into a single interface, thereby providing business applications with a single interface to perform datastore manipulation that would otherwise require two or more separate interfaces. In contrast, Gupta does not disclose or address the concept of datastore encapsulation. As noted in the title, abstract, and disclosure, Gupta is drawn to a method and apparatus for providing interoperability (also referred to as collaboration) between two or more applications – that is Gupta teaches how to improve efficiency and productivity by integrating related functionality across two or more applications (see, e.g., col. 2, lines 55-57). For example, a simplified business inter-operability function is receiving data from a first application and writing a portion of the data in a new format to a second application (see col. 5, lines 1-5). A more specific example is automatically generating an invoice and adding the amount to the accounts receivable entry in a financial system when a customer support application logs a new support call (see col. 13, lines 12-16 and Fig. 10). In summary, Gupta integrates functionality across applications in sharp contrast to Applicants' datastore encapsulation. More specifically, Applicants' invention is directed to increasing computing efficiency by allowing an application to easily access data from different types of datastores using a common interface whereas Gupta is directed to increasing computing efficiency by integrating functionality across two or more applications.

In an effort to better explain the differences between Applicants' invention and Gupta, explanation of a hypothetical combination of the two in a computing system may be helpful. In a combination with Gupta, Applicants' invention would apply to applications 70 in Fig. 1 of Gupta. In such a combination, Applicants' business component 5 in Applicants' Fig. 1 is equivalent to each of applications 70 in Fig. 1 of Gupta. To show more clearly the relationship, Applicants are providing herewith a copy of Fig. 1 of Gupta with handwritten additions showing the hypothetical combination with Applicants' invention. In such a combination, Applicants' invention would provide App 1 (i.e., Application 1) with a common interface 10 to a plurality of underlying datastores 15, 20A, and 25. For example, App 1 would be able to access customer information from a relational database, an object database, or combinations thereof using the common interface provided by Applicants' invention as described in the specification, see e.g., pages 6-9 and Fig. 2. Upon accessing such customer information according to Applicants' invention, App 1 could then share such information with Apps 2, 3, and 4 according to the interoperability and collaboration

means set forth in Gupta. For example, App 1 could be a customer service application which logs customer information (database accessibility provided according to Applicants' invention) following a service call and App 2 could be a billing application that generates an invoice using the customer information provided by App 1 (interoperability between App 1 and App 2 provided by Gupta, see e.g., col. 13, lines 12-16 and Fig. 10). Given the explanations above, it should be clear that Gupta does not disclose each and every element of Applicants' claimed invention, as these two are completely different computing solutions addressing completely different computing problems and the components thereof simply are not the same.

In attempting to equate the teaching of Gupta with each and every element of claim 1, the Examiner disregards Applicants' datastore encapsulation functionality and the express claim limitations drawn thereto. Instead, the Examiner incorrectly equates various parts of Gupta to elements of independent claim 1 as follows:

(a) a database wrapper (col. 19, lines 42-49, where "building object wrappers around the application data" is discussed) – The text relied upon by the Examiner relates to object representation module 1312, which is part of connecter 30 as shown in Fig. 3 and col. 18, lines 1-7. As explained above, an element of application connector 30, which provides an interface at the application level, is not the same or equivalent to Applicants' database wrapper, which provides an API for session and transaction management allowing access to database functionality (see page 16, line 15-16).

(b) a domain object factory – The Examiner relies upon col. 20, lines 15-24, which states in pertinent part "[a]n application connector is a component which operates like an 'object factory'" Assuming for sake of argument that connector 30 is an object factory, this is not the same or equivalent to Applicants' claimed domain object factory. As explained above, connectors 30 communicate at the application level with applications 70 and provide schema for interacting with other applications in the interchange server's object oriented model (see col. 4, lines 11-13). In Applicants' invention, implementation of a domain object is specific to a datastore (see page 11, line 12), and domain object factories contain

methods that allow for the creation and querying of domain objects, thereby creating domain objects and retrieving them from the datastore. Thus, connector 30 is not the same or equivalent to Applicants' domain object factory.

(c) a domain object (col. 4, lines 7-14) – As discussed previously, the passage relied upon by the Examiner relates to interoperability of two or more applications, and has nothing to do with underlying datastore accessibility, and in particular implementation of a domain object that is specific to a datastore. Furthermore, Applicants note that the Examiner is incorrectly equating a single element of Gupta (i.e., connector 30) with numerous of Applicants' separate and distinct elements.

(d) a datastore (col. 6, lines 65-67) – The text relied upon by the Examiner relates to repository 238, which is a part of interchange server 20, more specifically a part of service modules 208, as shown in Fig. 3 and col. 6, lines 40-50. Service modules 208 comprise a number of tools, including registry service 232, which coordinates the interchange server's startup processing and allows objects to find other objects by name (see col. 6, lines 58-60). In carrying out this function, registry service 232 stores meta-data associated with any component in repository 238 which can be accessed by other tools in the interchange server (see col. 6, lines 65-67). Based upon this description, respository 238 is not the same as or equivalent to Applicants' datastore, which is an underlying data storage device or medium accessible by a business application (see e.g., page 1, line 20 – page 2, line 1).

Given that Gupta does not disclose each and every element of independent claim 1, likewise it cannot disclose each and every element of claims 2-5 and 7-8 depending from claim 1. Furthermore, Gupta does not disclose each and every element of independent claim 9, and again disregards Applicants' datastore encapsulation functionality and the express claim limitations drawn thereto. The Examiner incorrectly equates various parts of Gupta to elements of independent claim 9 as follows:

(a) interfacing a database wrapper to a business component (col. 2, lines 1-5) – The text relied upon by the Examiner relates to connector 30, which serves as an interface to applications 70. In contrast, Applicants' are claiming a business component interfacing with an underlying database wrapper. In sum, an application interface is not the same as or equivalent to a database interface.

(b) implementing the database wrapper (col. 19, lines 42-44) – The text relied upon by the Examiner relates to object representation module 1312, which is part of connecter 30 as shown in Fig. 3 and col. 18, lines 1-7. As explained above, an element of application connector 30, which provides an interface at the application level, is not the same or equivalent to Applicants' database wrapper, which provides an API for session and transaction management allowing access to database functionality (see page 16, line 15-16).

(c) interfacing a domain object factory to the database wrapper (col. 4, lines 17-25) – The passage relied upon by the Examiner relates to connector 30 serving as a "virtual" object interface to the interchange server for the functionality of the application. Once again, the application level connector of Gupta is not the same as or equivalent to the database level elements set forth in Applicants' claims. As noted on page 16, lines 10, since a domain object factory implementation is specific to a given datastore, there must be a factory, herein referred to as a database wrapper, for the domain object factories.

(d) implementing the domain object factory (col. 5, lines 64 – col. 6, lines 9; col. 20, lines 15-24). The passage relied upon by the Examiner discloses the concept of implementation, but in an entirely different context from Applicants' invention. More specifically, the passage relates to the concept of implementing connectors and collaborations within the component-oriented object execution environment provided by interchange server 20. This is not the same as or equivalent to implementation of Applicants' domain object factory, wherein a

datastore-specific domain object factory is implemented for each datastore containing the requested domain object (see page 15, lines 15-17).

(e) interfacing a domain object to the domain object factory (col. 4, lines 17-25) – The passage relied upon by the Examiner relates to connector 30 serving as "virtual" object interface to the interchange server for the functionality of the application. Once again, the application level connector of Gupta is not the same as or equivalent to the database level elements set forth in Applicants' claims. As shown in Fig. 3 and on page 15, lines 10-13, domain object factories consist of an interface 45 that specifies create and find methods available on the domain object factory and separate implementations 50 and 55 specific to each datastore.

(f) implementing the domain object to retrieve data from a datastore (col. 20, lines 53-61) – The passage relied upon by the Examiner states in pertinent part that the connector communicates with the application and retrieves the data necessary to construct the requested object (for example, generate queries to a database). Again, the application level functionality of the connector in Gupta is not the same or equivalent to Applicants' database level implementation of a domain object that is specific to a datastore (see page 11, line 12).

Given that Gupta does not disclose each and every element of independent claim 9, likewise it cannot disclose each and every element of claims 10, 12, and 13 depending from claim 9. In sum, Gupta does not disclose datastore encapsulation as disclosed and claimed by Applicants, and therefore claims 1-5, 7-10, and 12-13 cannot be anticipated by Gupta. Given that claims 1-5, 7-10, and 12-13 were not otherwise rejected, Applicants respectfully submit that claims 1-5, 7-10, and 12-13 are in condition for allowance.

II. 103 Rejections

Claims 6 and 11 were rejected under 35 USC 103(a) as being unpatentable over Gupta in view of Carter (U.S. Pat. No. 5,878,419). Applicants respectfully submit that the combination of

Gupta and Carter does not establish a *prima facie* case of obviousness as to claims 6 and 11. According to MPEP 2142, three basic criteria must be met to establish a *prima facie* case of obviousness:

> First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. The teaching or suggestion to make the claimed combination and the reasonable expectation of success must both be found in the prior art, and not based on applicant's disclosure.

Assuming for sake of argument that the combination of Gupta and Carter is proper, the Examiner has nonetheless failed to establish a prima facie case of obviousness as such a combination does not teach or suggest all of the claim limitations. Claims 6 and 11 depend from and incorporate the limitations of independent claim 1. As discussed previously, Gupta does not disclose each and every element of claim 1, and more specifically does not disclose several recited elements related to datastore encapsulation. Carter is not cited by the Examiner for the purpose of providing these missing datastore encapsulation elements, and in any event does not do so even if relied upon for such. Furthermore, for the proposition that Gupta teaches a database as recited in claims 6 and 11, the Examiner relies upon col. 4, lines 3-6, which states:

> Platform in this case means any base software environment, including operating systems, databases (if necessary), and/or middleware.

When read in context, the passage relied upon by the Examiner has nothing to do with database encapsulation as claimed by Applicants. A database is disclosed in the context of a platform for operation of the interchange server – in other words, the invention according to Gupta may operate upon a database platform host. Clearly, such a database is not the same as or equivalent to Applicants' underlying database from which business applications access needed data. Furthermore, Carter does not make up for the deficiencies of the primary reference, Gupta. Assuming for the sake of argument that the combination of Gupta and Carter is proper (which Applicants respectfully submit is not the case), at most such combination teaches that a relational database may be used as a platform for operation of the interchange server, and this is not

equivalent to Applicants' claimed relational databases which are encapsulated according to the present invention. In sum, a prima facie case of obviousness has not been established as to claims 6 and 11.

Independent claim 14 was rejected under 35 USC 103(a) as being unpatentable over Gupta in view of McComb (U.S. Pat. No. 6,006,224). Independent claim 14 is likewise drawn to method for encapsulating a database. Again, the combination of Gupta and McComb (assuming without conceding such to be proper) does not disclose each and every element of independent claim 14, and in particular does not disclose the numerous elements reciting database encapsulation functionality. In attempting to equate the teaching of Gupta with each and every element of claim 14, the Examiner again disregards Applicants' database encapsulation functionality and the claim limitations drawn thereto. Instead, the Examiner incorrectly equates various parts of Gupta to elements of claim 14 as follows:

> Isolating a business component from specific implementations of a datastore – as discussed at length previously, Gupta does not teach, suggest, or even relate to such datastore isolation, but rather Gupta is drawn to a method and apparatus for providing interoperability (also referred to as collaboration) between two or more applications.
>
> (a) supplying a database wrapper (col. 19, lines 42-61) – The text relied upon by the Examiner relates to object representation module 1312, which is part of connecter 30 as shown in Fig. 3 and col. 18, lines 1-7. As explained above, an element of application connector 30, which provides an interface at the application level, is not the same or equivalent to Applicants' database wrapper, which provides an API for session and transaction management allowing access to database functionality (see page 16, line 15-16).
>
> (c) using the database wrapper to obtain a domain object factory (col. 2, lines 24-36) – The passage cited by the Examiner contains a recitation of the various elements of Gupta (e.g., modular application collaborator, interchange server, connector, application, service module) and the relationships there between, and it

is not explained which, if any, of these elements corresponds to the database wrapper and domain object factory recited in Applicants' claim. As explained previously, none of these elements from Gupta is the same as or equivalent to Applicants' recited elements.

(d) using the domain object factory to create a domain object (col. 4, lines 17-25; col. 5, line 64 – col. 6, line 9; col. 20, lines 15-24) – The first passage relied upon by the Examiner relates to connector 30 serving as a "virtual" object interface to the interchange server for the functionality of the application. Once again, the application level connector of Gupta is not the same as or equivalent to the database level elements set forth in Applicants' claims. As shown in Fig. 3 and on page 15, lines 10-13, domain object factories consist of an interface 45 that specifies create and find methods available on the domain object factory and separate implementations 50 and 55 specific to each datastore. The second and third passages relied upon by the Examiner disclose the concept of implementation, but in an entirely different context from Applicants' invention. More specifically, the passages relate to the concept of implementing connectors and collaborations within the component-oriented object execution environment provided by interchange server 20. This is not the same as or equivalent to implementation of a datastore-specific domain object factory for each datastore containing the requested domain object (see page 15, lines 15-17).

(e) converting the domain object from a persistent state to a transient state (col. 10, lines 12-24; col. 20, lines 31-34) – the passage relied upon by the Examiner relates to other services that may be provided by interchange server 20, specifically persistency service to allow for persistently storing objects or object state outside that of an application. At most, this passage discloses the concept of persistency broadly, as the passage provides no explanation of how such persistency functionality is incorporated into the invention disclosed by Gupta. Thus, such a broad disclosure of the concept of persistency is not the same as or

equivalent to the conversion of the domain object from a persistent state to a transient state as recited in Applicants' claims.

In sum, Gupta does not make obvious each and every element of independent claim 14, nor does McComb make up for the lack of teaching in Gupta. At most, McComb broadly teaches the concepts of using a database wrapper to begin a database session (i.e., connect) and end a database session (i.e., disconnect). Col. 9, lines 61-65 discloses return of pending messages associated with the terminated session, but does not disclose return of a domain object to the business component as claimed by Applicants in the context of database encapsulation. More importantly, however, the Examiner has not adequately explained how the teaching of McComb would be integrated into Gupta to yield Applicants' claimed invention, nor a basis for a reasonable expectation of success upon such integration. Rather, the patchwork of widely scattered and often cryptic passages relied upon by the Examiner as cited above shows that the Examiner appears to be impermissibly picking and choosing across the reference rather than reading the teaching of Gupta as a whole. When read as a whole, Gupta clearly relates to a completely separate computing problem and solution, and the hypothetical combination with McComb does not change the fundamental fact that the primary reference (Gupta) is substantially off point. It is impermissible within the framework of section 103 to pick and choose from any one reference only so much of it as will support a given position, to the exclusion of other parts necessary to the full appreciation of what such reference fairly suggests to one of ordinary skill in the art. See, e.g., In re Wesslau, 147 USPQ 391, 393 (1965). When read as a whole, neither Gupta alone or in combination with Carter or McComb makes obvious Applicant's claimed datastore encapsulation invention, and thus Applicant respectfully submits that claims 6, 11, and 14 are in condition for allowance.

III. Conclusion

The Commissioner is hereby authorized to charge payment of any further fees associated with any of the foregoing papers submitted herewith, or to credit any overpayment thereof, to Deposit Account No. 50-1515 , Conley, Rose & Tayon.

Attorney Docket No. IDF 1415 (4000-00900)

Applicants respectfully submit that the present application is in condition for allowance. If the Examiner has any questions or comments or otherwise feels it would be helpful in expediting the application, he is encouraged to telephone the undersigned at (972) 731-2288.

Respectfully submitted,
CONLEY, ROSE & TAYON, P.C.

Date: September 19, 2002

5700 Granite Parkway, Suite 330
Plano, Texas 75024
Telephone: (972) 731-2288
Facsimile: (972) 731-2289

Rodney B. Carroll
Reg. No. 39,624

ATTORNEY FOR APPLICANT

84706.01/4000.00900

US005913061A

United States Patent [19]

Gupta et al.

[11] Patent Number: 5,913,061

[45] Date of Patent: Jun. 15, 1999 + 1 yr June 15, 2000

[54] MODULAR APPLICATION COLLABORATION

[75] Inventors: Prashant Gupta, Monterey; Katrina A. Mellen-Garnett, Hillsborough, both of Calif.

[73] Assignee: CrossRoads Software, Inc., San Mateo, Calif.

[21] Appl. No.: 08/780,593

[22] Filed: Jan. 8, 1997 — 102(e)

[51] Int. Cl.[6] G06F 9/40

[52] U.S. Cl. 395/680; 395/671

[58] Field of Search 395/680, 671, 395/682

Current 709/101

[56] References Cited

U.S. PATENT DOCUMENTS

5,732,270 3/1998 Foody et al. 395/683

OTHER PUBLICATIONS

Orfali et al., "The Essential Client/Server Survival Guide", second edition, Wiley Computer Publishing, pp. 1–676, 1996.

Betz, Mark, "Interoperable objects: laying the foundation for distributed-object computing" Dr. Dobb's Journal, v19, n11, p18(13), Oct. 1994.

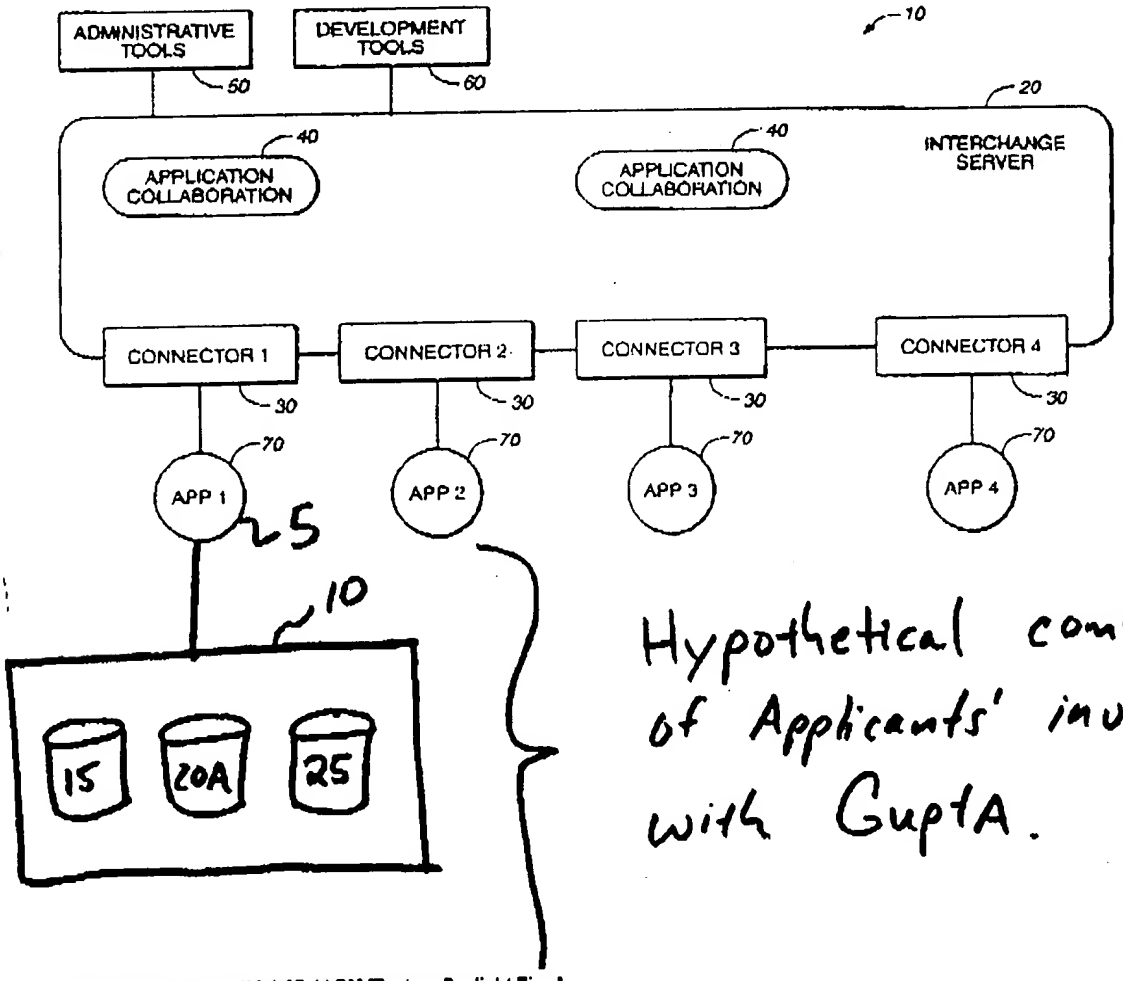
Dolgicer, Max, "A Formal Look at Tuxedo", Data Communications, p. 33; vol. 22, No. 12, Sep. 1993.

Primary Examiner—Alvin E. Oberley
Assistant Examiner—St. John Courtenay, III
Attorney, Agent, or Firm—Fish & Richardson P.C.

[57] ABSTRACT

A modular application collaborator for providing interoperability between applications including a plurality of connectors for communicating with a like plurality of applications and an interchange server. The interchange server includes an application collaboration module and a service module. The service module transfers messages between connectors and the application collaboration module. The application collaboration module defining the interoperability between two or more applications and includes a trigger and a transaction responsive to the trigger. The trigger is activated upon receipt of data from one or more connectors resulting in the transaction delivering data to one or more connectors for transfer to an associated application.

8 Claims, 15 Drawing Sheets



Hypothetical combination of Applicants' invention with GuptA.